R-Kurs Labb 1

Edvin Magnusson

2022-09-14

R Markdown

Uppgift 1

Creating a vector function

```
# Clear the enviroment
 rm(list = ls())

my_num_vector <- function(){
  x <- c(log10(11),cos(pi/5),exp(pi/3),(1173%%7)/19)
  return(x)
}

my_num_vector()
```

```
## [1] 1.0413927 0.8090170 2.8496539 0.2105263
```

Uppgift 2

Create a function called mult_first_last() with the argument vector. The function shall return the product of the first and last element in vector.

```
mult_first_last<- function(vektor){

  result<- vektor[1]*tail(vektor,1)

  return(result)

}

mult_first_last(vektor = c(3,1,12,2,4))
```

```
## [1] 12
```

```
mult_first_last(vektor = c(3,1,12))
```

```
## [1] 36
```

Uppgift 3

Create a function called orth_scalar_prod() which calculate the scalar product between two vectors, a and b, in an orthonormal base. The scalar product is calculated in the following way:

```
orth_scalar_prod<- function (a,b){

  result<- a%*%b
  return(result)

}

orth_scalar_prod(a = c(3,1,12,2,4), b = c(1,2,3,4,5))
```

```
##      [,1]
## [1,]   69
```

```
orth_scalar_prod(a = c(-1, 3), b = c(-3, -1))
```

```
##      [,1]
## [1,]    0
```

Uppgift 4

```
lukes_father <- function(name){

  answer<- "I am your father."


  cat(name,answer,sep = ",")

}

lukes_father(name = "Luke")
```

```
## Luke,I am your father.
```

```
lukes_father(name = "Edd")
```

```
## Edd,I am your father.
```

Uppgift 5

Approximate the number e formula sum(1/n!)

```
# The trick is that when n=0 and 1 the answer is 1+1

approx_e<- function(N){
```

```
  i<- 2:N
  factorial_part <- 1/factorial(i)

  answer<- 1+1+sum(factorial_part)
  return(answer)


}

approx_e(N = 2)
```

```
## [1] 2.5
```

```
approx_e(N = 4)
```

```
## [1] 2.708333
```

```
approx_e(N=7)
```

```
## [1] 2.718254
```

With N=7 the approximation of e is correct with four decimals.

Question 6

Create a function called filter_my_vector() with the arguments x and geq. The function should take a vector x and set all values greater than or equal to geq to missing value (NA).

```
filter_my_vector <- function(x,geq){

  x[x>=geq]<-NA
  return(x)
}


filter_my_vector(x = c(2, 9, 2, 4, 102), geq = 4)
```

```
## [1]  2 NA  2 NA NA
```

Question 7

create a magic matrix

```
my_magic_matrix<- function (){

  x<-c(4,9,2,3,5,7,8,1,6)
  Mat<- matrix(x,nrow = 3,ncol = 3,byrow = TRUE)
  return(Mat)

}
my_magic_matrix()
```

```
##      [,1] [,2] [,3]
## [1,]    4    9    2
## [2,]    3    5    7
## [3,]    8    1    6
```

```
# every row and column adds up to 15
```

Question 8

Create a function called calculate_elements(A) that can take a matrix of an arbitrary size and calculate the number of elements in the matrix.

```
Calculate_elements <- function(A){

  Result<- nrow(A)*ncol(A)
  return(Result)

}

mat<-my_magic_matrix()

Calculate_elements(A=mat)
```

```
## [1] 9
```

```
new_mat <- cbind(mat, mat)
Calculate_elements(A = new_mat)
```

```
## [1] 18
```

Uppgift 9

Create a function called row_to_zero(A, i) that can take a matrix of an arbitrary size and set the row indexed with i to zero.

```
row_to_zero<-function(A,i){

  A[i,]<-0
  return(A)
}


row_to_zero(A=mat,i=3)
```

```
##      [,1] [,2] [,3]
## [1,]    4    9    2
## [2,]    3    5    7
## [3,]    0    0    0
```

```
row_to_zero(A=mat,i=1)
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    3    5    7
## [3,]    8    1    6
```

Uppgift 10

Create a function called add_elements_to_matrix() with parameters A, x, i, j. The function should take a matrix A of an arbitrary size and add the value x to the parts of A indexed by row(s) i and column(s) j.

```
add_elements_to_matrix<-function(A,x,i,j){

  A[i,j]<- A[i,j]+x
  return(A)

}

add_elements_to_matrix(A = mat, x = 10, i = 2, j = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    4    9    2
## [2,]    3    5   17
## [3,]    8    1    6
```

```
add_elements_to_matrix(A = mat, x = -2, i = 1: 3, j = 2: 3)
```

```
##      [,1] [,2] [,3]
## [1,]    4    7    0
## [2,]    3    3    5
## [3,]    8   -1    4
```

Question 11

```
my_magic_list<- function(){

my_text<-"My own list"
my_vector<-my_num_vector()
my_matrix<-my_magic_matrix()

listan<- list(info= my_text,my_vector,my_matrix)
return(listan)
}

my_magic_list()
```

```
## $info
## [1] "My own list"
```

```
##
## [[2]]
## [1] 1.0413927 0.8090170 2.8496539 0.2105263
##
## [[3]]
##      [,1] [,2] [,3]
## [1,]    4    9    2
## [2,]    3    5    7
## [3,]    8    1    6
```

Question 12

Create a function that will take a list x (that must contain one element with name info) and change this element to the text argument given by text.

```
change_info <- function(x, text) {
  x[1] <- text
  return(x)
}

change_info(x=my_magic_list(),text="some new info")
```

```
## $info
## [1] "some new info"
##
## [[2]]
## [1] 1.0413927 0.8090170 2.8496539 0.2105263
##
## [[3]]
##      [,1] [,2] [,3]
## [1,]    4    9    2
## [2,]    3    5    7
## [3,]    8    1    6
```

Question 13

Create a function called sum_numeric_parts() that will take a list x and sum together all numeric elements in this list.

```
sum_numeric_parts <-function(x){
  sum_num<- sum(as.numeric(unlist(x)),na.rm = TRUE)
  return(sum_num)
}


a_list<-my_magic_list()

sum_numeric_parts(x=a_list)
```

```
## Warning in sum_numeric_parts(x = a_list): NAs introduced by coercion
```

```
## [1] 49.91059
```

```
sum_numeric_parts(x=a_list[2])
```

```
## [1] 4.91059
```

Question 14

Create a function that will take a list x and add a new list element with the name note. This new element should contain text from the note parameter.

```
add_note <- function(x,note){
  New_list_element<- note
  old_list<- x
  ret<- list(info= old_list,note=New_list_element)
  return(ret)
}

a_list <- my_magic_list()
add_note(x = a_list, note = "This is a magic list!")
```

```
## $info
## $info$info
## [1] "My own list"
##
## $info[[2]]
## [1] 1.0413927 0.8090170 2.8496539 0.2105263
##
## $info[[3]]
##      [,1] [,2] [,3]
## [1,]    4    9    2
## [2,]    3    5    7
## [3,]    8    1    6
##
##
## $note
## [1] "This is a magic list!"
```

Question 15

Create a function that generates a data.frame

```
my_data.frame<-function(){
 id <- c(1,2,3)
     name <- c("John", "Lisa", "Azra")
     income <- c(7.30,0.00, 15.21)
     rich <- c(FALSE, FALSE, TRUE)
     df <- data.frame(id = id, name = name, income = income, rich = rich)
  return(df)

}

my_data.frame()
```

```
##   id name income  rich
## 1  1 John   7.30 FALSE
## 2  2 Lisa   0.00 FALSE
## 3  3 Azra  15.21  TRUE
```

Question 16

Sorting a data.frame by one variable

```
sort_head<- function(df,var.name,n){

  sorting<- head(sort(df[,var.name],decreasing=TRUE),n)
  return(sorting)

}

sort_head(df=iris,var.name="Petal.Length",n=5)
```

```
## [1] 6.9 6.7 6.7 6.6 6.4
```

```
# Alternativley

sort_head2 <- function(df,var.name,n){
  sort2<-head(df[order(df[,var.name],decreasing = TRUE),],n)
  return(sort2)

}

sort_head2(df=iris,var.name="Petal.Length",n=12)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 119          7.7         2.6          6.9         2.3 virginica
## 118          7.7         3.8          6.7         2.2 virginica
## 123          7.7         2.8          6.7         2.0 virginica
## 106          7.6         3.0          6.6         2.1 virginica
## 132          7.9         3.8          6.4         2.0 virginica
## 108          7.3         2.9          6.3         1.8 virginica
## 110          7.2         3.6          6.1         2.5 virginica
## 131          7.4         2.8          6.1         1.9 virginica
## 136          7.7         3.0          6.1         2.3 virginica
## 101          6.3         3.3          6.0         2.5 virginica
## 126          7.2         3.2          6.0         1.8 virginica
## 103          7.1         3.0          5.9         2.1 virginica
```

Question 17

Create a compared to median function

```
add_median_variable<-function(df,j){

  Median<- median(df[,j])
```

```
  df$Compared_to_median<- ifelse(df[,j]==Median,c("Median"),ifelse(df[,j]<Median,c("Smaller"),c("Greate

  return(df)

}

head(add_median_variable(df = faithful, 1), n = 12)
```

```
##    eruptions waiting Compared_to_median
## 1      3.600      79            Smaller
## 2      1.800      54            Smaller
## 3      3.333      74            Smaller
## 4      2.283      62            Smaller
## 5      4.533      85            Greater
## 6      2.883      55            Smaller
## 7      4.700      88            Greater
## 8      3.600      85            Smaller
## 9      1.950      51            Smaller
## 10     4.350      85            Greater
## 11     1.833      54            Smaller
## 12     3.917      84            Smaller
```

Question 18

```
analyze_columns<- function(df,j){

  Mean<- apply(df[,j],2,mean)
  Median<- apply(df[,j],2,median)
  SD<- apply(df[,j],2, sd)

  Statistics <- rbind(Mean,Median,SD)

  Corr <- cor(df[,j])

  Listan<- list(Statistics,Correlation_Matrix=Corr)

  return(Listan)

}

analyze_columns(df = faithful, 1:2)
```

```
## [[1]]
##        eruptions  waiting
## Mean    3.487783 70.89706
## Median  4.000000 76.00000
## SD      1.141371 13.59497
##
## $Correlation_Matrix
##             eruptions   waiting
```

```
## eruptions 1.0000000 0.9008112
## waiting   0.9008112 1.0000000
```

```
analyze_columns(df = iris, c(1,3))
```

```
## [[1]]
##        Sepal.Length Petal.Length
## Mean      5.8433333     3.758000
## Median    5.8000000     4.350000
## SD        0.8280661     1.765298
##
## $Correlation_Matrix
##              Sepal.Length Petal.Length
## Sepal.Length    1.0000000    0.8717538
## Petal.Length    0.8717538    1.0000000
```

```
# Alternatively for better formatting you can use lapply
# It returns elements as list objects from vectors etc

analyze_columns <- function(df, j){

    mean <- data.frame(lapply(df[,j], mean))
    median <- data.frame(lapply(df[,j], median))
    sd <- data.frame(lapply(df[,j], sd))
    rb <- as.data.frame(rbind(mean, median, sd))
    cor <- cor(df[,j])
    out <- list(c(rb), correlation_matrix = cor)


    return(out)
    }


analyze_columns(df = faithful, 1:2)
```

```
## [[1]]
## [[1]]$eruptions
## [1] 3.487783 4.000000 1.141371
##
## [[1]]$waiting
## [1] 70.89706 76.00000 13.59497
##
##
## $correlation_matrix
##           eruptions   waiting
## eruptions 1.0000000 0.9008112
## waiting   0.9008112 1.0000000
```

```
analyze_columns(df = iris, c(1,3))
```

```
## [[1]]
## [[1]]$Sepal.Length
```

```
## [1] 5.8433333 5.8000000 0.8280661
##
## [[1]]$Petal.Length
## [1] 3.758000 4.350000 1.765298
##
##
## $correlation_matrix
##              Sepal.Length Petal.Length
## Sepal.Length    1.0000000    0.8717538
## Petal.Length    0.8717538    1.0000000
```

Question 19

Trace of a matrix

```
A <- matrix(2:5, nrow = 2)
B <- matrix(1:9, nrow = 3)
C <- matrix(9:-6, nrow = 4)

matrix_trace <- function (X){
  X[upper.tri(X)] <- 0
  X[lower.tri(X)] <- 0
  sum <- sum(X)
  return(sum)
}

matrix_trace(A)
```

```
## [1] 7
```

```
# Alternatively

matrix_trace1 <- function (X){
  sum_diag <- sum(diag(X))
  return(sum_diag)
}

matrix_trace1(A)
```

```
## [1] 7
```